Monday, August 1, 1977

Dear Ron,

*file under "Schoepf"

Sorry I took so long to answer. As far as I know, your $100 should be safe. I haven't tried out your number; I'm assuming it was constructed as specified in your paper.

I like the scheme; it is pretty. It looks like it should work. It would be a pleasant surprise for factoring to have a practical application.

The good methods of factoring:

1) My method. It is unproven, since it hasn't cracked any numbers yet. Timing is $O(e^{\sqrt{\log N \log\log N}})$.

2) Continued fractions, as explained in Knuth v2 p351-354. Timing is $O(e^{\sqrt{2 \log N \log\log N}})$. Inferior to my method for $N>10^{20}$.

2a) A variant on (2), using Pollard's x^2+1 method to factor the Vs in step E3 on p352 of Knuth. Timing is $O(e^{\sqrt{1.5 \log N \log\log N}})$. This gives an asymptotic improvement to (2), but the crossover point is probably $N>10^{30}$. So either (1) or (2) will always be better than (2a).

3) Pollard's x^2+1 method. Applicable when N has a small factor f less than about 10^{18}. Timing $O(\sqrt{f})$.

4) Shanks's method, using quadratic forms. I don't understand this method. He claims it is $\sqrt[4]{N}$, but I think it's $\sqrt[5]{N}$. He did not agree to $\sqrt[5]{N}$ when I wrote him.

5) Various $\sqrt[3]{N}$ methods, which are interesting but not competitive in the number range of interest.

6) The difference of squares sieving method, applicable when N=fg and f/g is near a rational number r/s with small r,s. The Lehmers have a machine which carries out the sieving very rapidly.

7) Pollard's method based on creating a highly composite number (mod N), which will have a common factor f with N. Works if f-1 has only one moderately large divisor d, say $d<10^{10}$. Timing $O(d)$.

7a) Variations on (7) that depend on f+1 (or f^2+f+1, or other cyclotomic polynomials in f) instead of f-1.

8) Various methods applicable when N is the sum or difference of exact powers.

From your paper, I assume you are covered on methods 3,6,7. That a random number would satisfy 8 is very improbable, but a subroutine to check this out would be easy to add to the program that creates composites. Method 7a is a problem for your scheme. A possible solution: Since we know f, examine several cyclo(f) with method 3, and reject f if any cyclo(f) factors too easily. Another possibility: If f is large, the odds of f+1 having only moderate size divisors are poor; the odds for f^2+f+1 are worse, etc. Just make f big and take your chances. A third idea: Use a modulus made of several large prime factors; then even if one of the factors is compromised, the modulus is uncompromised.

My current estimate of the time required to factor F8 is 10000 hours of computer time (on a KA10). This makes the constant in my timing formula 2.4 microseconds. Plugging in $N=10^{128}$, I get a time of 50000 years. Of course there are faster computers than the KA10, perhaps by a factor of 300. And the technology keeps getting better. But your number should be safe for many years.

The estimated time for my method to factor variously sized numbers on a KA10:

N	10^{50}	2^{256}	10^{100}	10^{128}	2^{512}
$e^{\sqrt{\log N \log\log N}}$	1.4×10^{10}	1.5×10^{13}	2.3×10^{15}	6.0×10^{17}	6.7×10^{19}
KA10 time	10 hr	1.2 yr	180 yr	50000 yr	5 Myr

The chances of a better factoring method coming along are unknown, but certainly not small. Several advances have been made in the past few years. I have had a lot of ideas that turned out to be dead ends, but a few were winners. I haven't given up looking. Several other clever people are looking too. If your encryption scheme is adopted, it will attract more clever people to the quest.

My method: Suppose $Q=\lceil\sqrt{N}\rceil$. Search for instances of

$$(Q+A)(Q+B)-N = \pm\prod P_i$$

where A and B are small, and P_i are primes of moderate size. For my factorization of F8, $Q=2^{128}$, $-2^{18} \le A \le 2^{18}$, $A \le B \le A+10^6$. All but 2 of the P_i are less than 10^6; the two largest P_i are less than 2^{30}. The solutions are combined in the same way as the continued fraction method to produce a nontrivial solution of $Z^2=1 \pmod N$. Then GCD(N,Z-1) is a factor of N. The efficiency of the method comes from fixing A, and doing all the Bs in one calculation: The linear progression $(Q+A)(Q+B)-N$ is formed in core memory, and then sieved to remove all divisors less than 10^6. Any leftover pieces less than 2^{60} are tested for primality, and if composite are factored by a continued fraction routine. The time for a particular A is about 8.5 minutes. Each A produces

$8,500 hrs
= 300+ days

about 16 solutions with $Pi<2^{27}$. I estimate that about 10^6 solutions are necessary, and plan to search about 60,000 values of A. Presently I have processed 20000 values.

The theory of the running time is complicated, depending on Dickman's function rho(x). Rho(x) is the fraction of numbers N such that all prime factors of N are less than $\sqrt[x]{N}$.

My method should work equally well on all numbers, including those with more than two prime factors.

Some comments on your paper: In view of my method, your minimal numbers should be larger.

On p.3, you claim that "determining whether a given integer x is prime or not can also be performed efficiently, even if the integer x is several hundred bits long." This is not strictly true, unless the factorization of x-1 (or some other cyclotomic polynomial in x) is available. This is not a handicap to your method, since many methods of making large primes involve adding 1 to a number of known factorization.

Again on p.3, you put double quotes around the word "proof" after equation 4. This suggests that the proof is only probabilistic. This is false: finding an A such that $A^N \not\equiv A \pmod N$ is a perfectly valid proof that N is composite.

On p.4, I would suggest a notation change. Using p and q for primes is quite standard, but you shouldn't continue assigning letters in the same sequence to nonprimes. Perhaps m would be a good symbol for pq, and e for the public exponent you have called s. Another literary suggestion: name your protagonists, perhaps Adolf and Bertholt or somesuch. This would reserve isolated letters for mathematical quantities.

Again on p.4, line 17, a sentence begins "When A puts r and s ...". This sentence would be improved by deleting everything before the first comma.

On p.4, line 28: GCD(p-1,r-1) = GCD(q-1,r-1) = GCD(p-1,q-1). It is probably wise to also check GCD(p-w1,q-w2) where w1 and w2 are roots of unity. Or, check GCD(C1(p),C2(q)) where C1 and C2 are cyclotomic polynomials.

The method of generating primes is good, but should include enough random elements that a potential cryptographer could not hope to search the space of probable primes.

On page 5, the bottom paragraph is devoted to showing the invertibility of EA, even if M is not relatively prime to r. No modification to the decoder is needed to handle this case. As long as r is squarefree, then for all M, $M^{phi(r)+1} = M \pmod r$.

Proof: Let p|r. Then p-1|phi(r), so $M^{phi(r)+1} = M \pmod p$. As r has no square factor, r=LCM(p: p|r), so $M^{phi(r)+1} = M \pmod r$.

I wrote to the University of Waterloo to obtain Gary Miller's technical report (your reference 4), but they said it was out of print and no more would be printed. Could you send me a xerox of yours? Thanks.

all p s.t. p divides n

Rich Schroeppel

Rich Schroeppel

cc Marti Hellman, Witt-Diffie, Bill Gosper